

Online CAT nástroj pro překlad textů

OnLine Computer Aided Translation Tool

Bc.Marcela Uličná

Diplomová práce
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

*** nescannované zadání str. 1 ***

*** nescannované zadání str. 2 ***

ABSTRAKT

Cílem diplomové práce je navrhnout online nástroj pro překlad textů. Teoretická část se nejprve zaměřuje na analýzu dostupných překládacích nástrojů, poté čtenáře seznamuje s nepříliš známými webovými technologiemi, které se v poslední době začínají stále více prosazovat. Text je doplněn praktickými ukázkami jednoduchých kódů. Praktická část se zabývá návrhem a implementací aplikace. V závěru je zmíněno předpokládané rozšíření nástroje.

Klíčová slova:

webové technologie, databáze, překladač, online překládací nástroj, MVC architektura, MongoDB, Node, Backbone

ABSTRACT

The aim of this thesis is designing of online computer aided translation tool. Theoretical part is firstly focused on analysis of available translation tools, after that it introduces the readers with little known web technologies that are increasingly to assert nowadays. Text is accompanied by practical demonstrations of simple codes. Practical part describes designing and implementation of application. There is mentioned the expected extensions of tool in the conclusion.

Keywords:

web technologies, databases, translator, online translation tool, MVC architecture, MongoDB, Node, Backbone

Na tomto místě bych ráda poděkovala vedoucímu diplomové práce panu Ing. Petru Šilhavému, Ph.D za jeho podnětné návrhy a cenné připomínky při vedení mé práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, 5.5.2013

.....
podpis diplomanta

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 NÁSTROJ PRO PŘEKLAD TEXTŮ	11
1.1 VÝZNAM PŘEKLÁDACÍCH NÁSTROJŮ	11
1.2 ANALÝZA DOSTUPNÝCH NÁSTROJŮ	11
1.2.1 Desktopové aplikace	11
1.2.2 Webové aplikace	15
1.3 VYHODNOCENÍ ANALÝZY	17
2 WEBOVÉ APLIKACE	18
2.1 KLIENTSKÁ ČÁST	19
2.1.1 Model v Backbone.js	19
2.1.2 View v Backbone.js	21
2.1.3 Controller	22
2.2 SERVEROVÁ ČÁST	22
2.3 DATABÁZE	23
2.3.1 Schema	24
2.3.2 Vyhledávání	25
2.3.3 CRUD operace	26
2.4 SHRNUÍ	27
II PRAKTICKÁ ČÁST	28
3 CÍLE APLIKACE	29
3.1 POŽADAVKY	29
3.1.1 Funkční požadavky	29
3.1.2 Nefunkční požadavky	29
3.2 NÁVRH DATABÁZE	30
3.2.1 User	30
3.2.2 File	30
3.2.3 FileSegment	31
3.2.4 SegmentDatabase	31
3.2.5 Language	33
3.3 ARCHITEKTURA APLIKACE	34
3.3.1 Struktura aplikace	34
3.3.2 Serverová část aplikace	34

4	UŽIVATELSKÉ ROZHRANÍ APLIKACE.....	39
4.1	REGISTROVÁNÍ A PŘIHLÁŠENÍ DO APLIKACE	39
4.2	HLAVNÍ STRÁNKA APLIKACE	39
4.3	STRÁNKA PRO NAHRÁVÁNÍ SOUBORŮ.....	40
4.4	PŘEKLADOVÁ STRÁNKA	41
5	DALŠÍ ROZŠÍŘENÍ	44
6	SYSTÉMOVÉ POŽADAVKY	45
6.1	INSTALACE NODE.JS	45
6.2	INSTALACE MONGODB.....	45
6.3	SPUŠTĚNÍ APLIKACE.....	45
	ZÁVĚR.....	47
	CONCLUSION	48
	SEZNAM POUŽITÉ LITERATURY	50
	SEZNAM OBRÁZKŮ	51
	SEZNAM PŘÍLOH	52

ÚVOD

Každý z nás se určitě ocitl v situaci, kdy potřeboval přeložit text z cizího jazyka do jazyka, kterému rozumí nebo naopak. Pak jistě sáhl po nějakém online nástroji, ať už to byl obyčejný slovník nebo nějaký překladač. Pro pochopení textu je to v mnoha případech dostačující.

Avšak pokud bychom se chtěli věnovat profesionálnímu překladu textu, budeme muset sáhnout po jiném řešení, většinou v podobě desktopových aplikací, kterých existuje celá řada. Ne vždy je to však ideální řešení, ať už je to cena takových nástrojů, nebo nutnost instalace a z toho plynou i větší nároky na systém.

Účelem webových překládacích nástrojů by mělo být zajištění příjemného uživatelského rozhraní pro překlad textů spojené s kvalitní databází přeložených spojení. Jak se můžeme sami přesvědčit, jen málokterý nástroj splňuje alespoň částečně tyto požadavky.

Předmětem této práce je vyvinout nástroj, který by využíval výhody přístupu k aplikaci přes webový prohlížeč. Cílem bylo navrhnout takové prostředí, které by umožňovalo každému uživateli v první řadě příjemnou práci s aplikací a také dostatečný přehled o všech svých dokumentech. Důležitou funkcí systému je, že uživatel se sám podílí na vytváření databáze přeložených vět neboli segmentů. Tím se zajistí přijatelná kvalita překladů.

Před samotnou realizací bylo nejprve potřeba zvolit vhodné technologie. I když se nabízí využití klasických webových technologií, v této práci byl dán prostor novějším technologiím vycházející z klientského skriptovacího jazyka JavaScriptu. Výhodou tohoto přístupu je především použití jednoho programovacího jazyka a tím pádem i rychlejší vývoj aplikace. Pro ukládání dat byl zvolen databázový systém MongoDB, jeden z předních NoSQL systémů.

Pro správnou funkčnost aplikace je potřeba zajistit bezproblémový chod na nejpoužívanějších prohlížečích (Internet Explorer, Firefox, Chrome).

I. TEORETICKÁ ČÁST

1 NÁSTROJ PRO PŘEKLAD TEXTŮ

1.1 Význam překládacích nástrojů

Hlavní účel nástrojů pro překlad textů z jednoho jazyka do druhého je především usnadnit překladateli práci na překladu textu. Díky těmto nástrojům se jejich práce stává efektivnější. Pokud navíc nástroj obsahuje dostatečně velkou databázi přeložených výrazů nebo má zabudované slovníky pro dané jazyky, odrazí se to pozitivně na kvalitě překladu.

Uživatele těchto nástrojů můžeme rozdělit do dvou skupin. První skupinu tvoří běžní uživatelé, kteří potřebují přeložit větší množství textu a využívají připojené slovníky. Většinou se spokojí s pochopením textu a nezáleží jim až tak moc na kvalitě překladu. Druhou skupinou jsou pak překladatelé, kteří sami překládají dané texty a slovníky využívají jen jako pomůcku pro překlad.

1.2 Analýza dostupných nástrojů

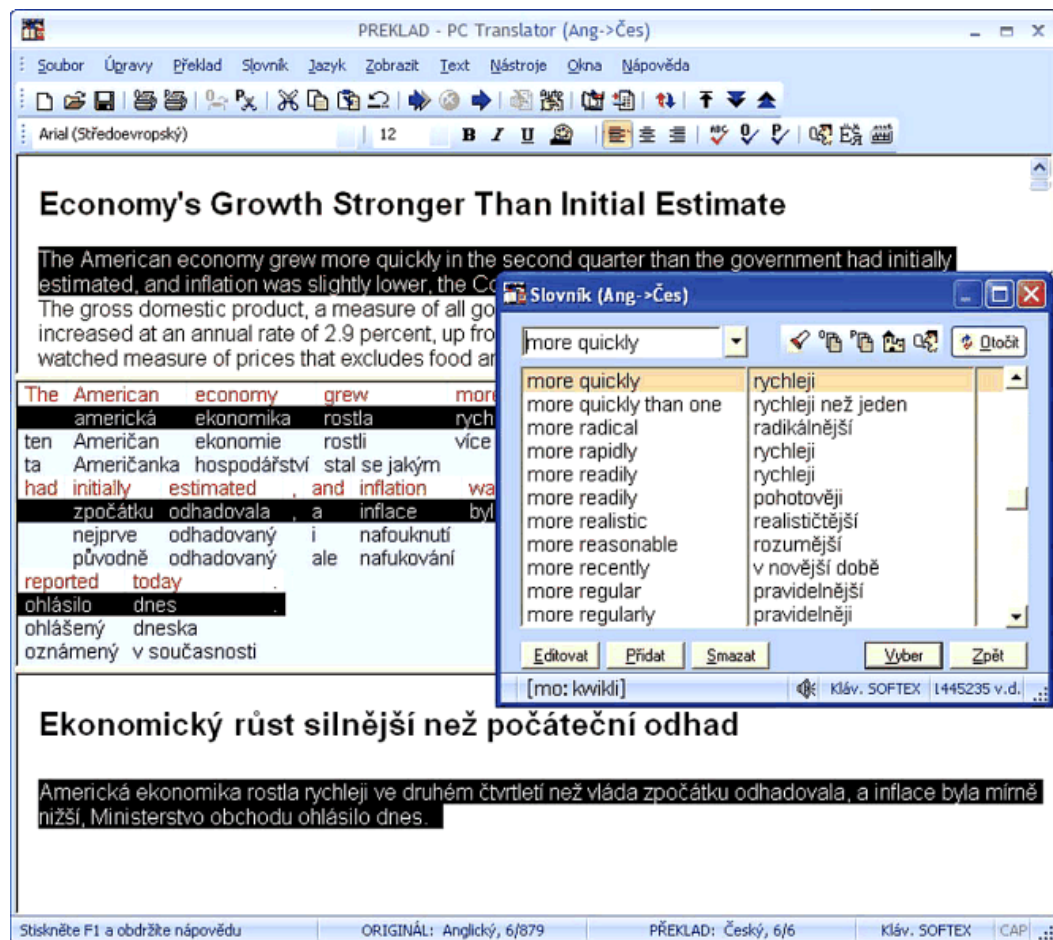
V dnešní době již existuje celá řada překládacích nástrojů, a to jak desktopové, tak i webové. Každý nástroj však poskytuje jinou úroveň překladu.

1.2.1 Desktopové aplikace

U desktopových aplikací jsme si zvykli na poměrně vysokou kvalitu překladu. Většina z nich umožňuje interaktivní komunikaci s vestavěnou databází přeložených výrazů a slovníkem. Slovníky jsou typicky rozdělené podle oborů a obsahují i spojení a idiomy, které se v daném jazyce nacházejí. Následuje výčet vybraných překládacích nástrojů:

- Slovník a překladač PC Translator[1]: tento nástroj obsahuje překladač textů pro překlad do češtiny a z češtiny do cizího jazyka i multioborový slovník. Slovníkové výrazy jsou navíc namluvené rodilými mluvčími. Texty lze do překladače přímo psát nebo vložit ze schránky, ten může být potom překládán zcela automaticky nebo

po větách. Nástroj je vhodný pro začátečníky i pro pokročilé uživatele a velice kladně hodnocený¹.



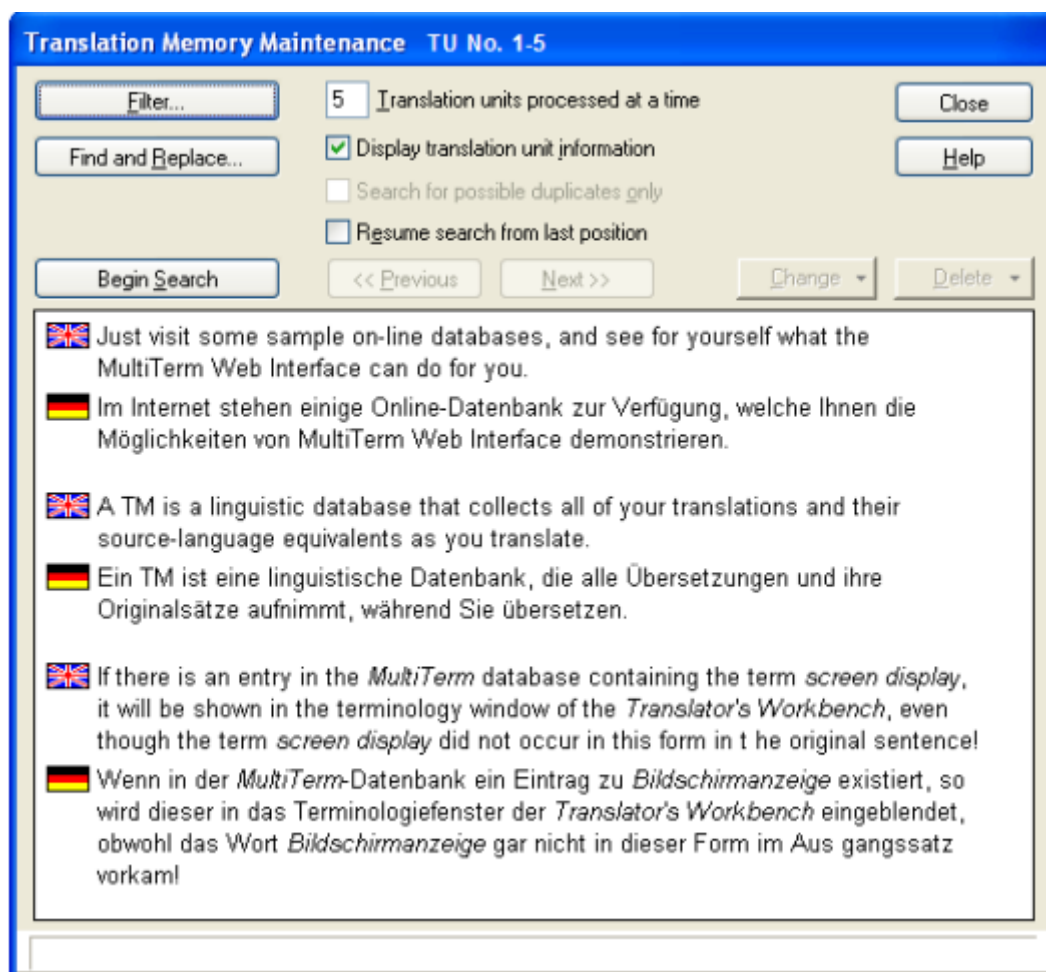
Obr. 1. Uživatelské rozhraní softwarového nástroje PC Translator

- SDL Trados 2007[2]: jedná se o komerční nástroj, používaný překladatelskými společnostmi. Je to jeden z nejznámějších a nejpoužívanějších nástrojů pro překlad textu profesionálními překladateli. Umožňuje překlad z libovolného jazyka do jiného. Dále podporuje konverzi celé řady formátů. Neobsahuje klasický slovník, je to spíše sofistikovaný databázový systém postavený na tzv. *translation memory*, což je metoda, která umožňuje ukládání a znovupoužití (recyklaci) přeložených částí. Uživatel si tak sám vytváří databázi překladů pro daný jazyk. Navíc každý

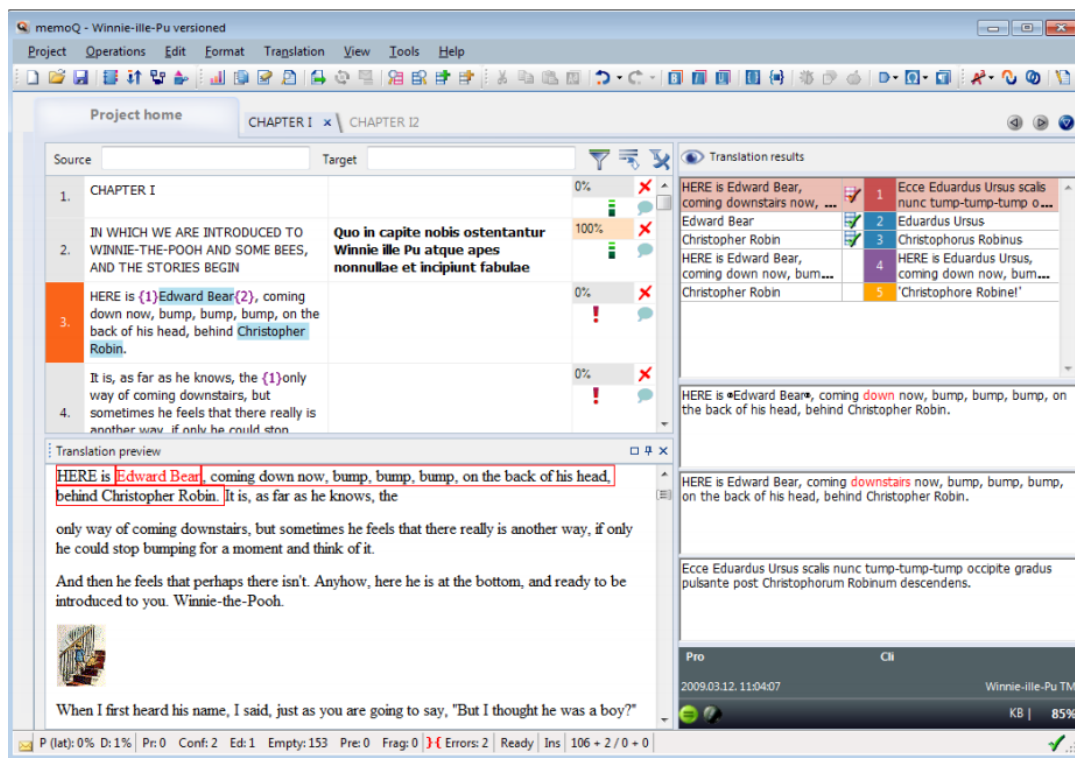
¹ Recenze softwarového nástroje PC Translator je k dispozici na webové stránce

<http://pcworld.cz/software/recenze-pc-translator-2009-prekladejte-chytre-7581>

nabízený výraz je ohodnocený na základě podobnosti s právě překládaným výrazem a překladatel ihned vidí, jestli je daný výraz vhodný.



Obr. 2. Translation Memory interface aplikace SDL Trados 2007

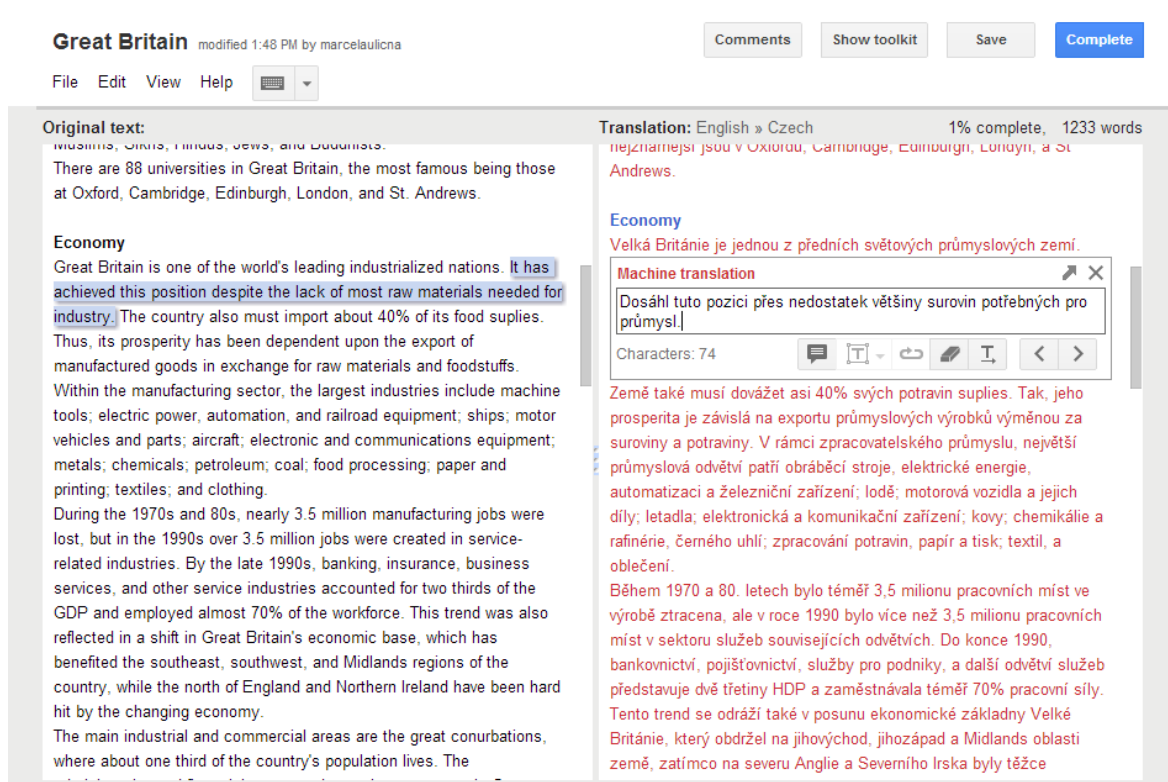


Obr. 4. Uživatelské rozhraní nástroje memoQ

Mezi výhody desktopových aplikací můžeme zařadit dostatečnou rychlost, protože vše běží lokálně na počítači uživatele a také skutečnost, že počítač nemusí být nutně připojený k Internetu. Naopak nevýhoda tohoto řešení spočívá v nemožnosti sdílení databáze překladu nebo slovníku a uživatel tak spoléhá pouze na vestavěný slovník nebo vlastní databázi přeložených výrazů.

1.2.2 Webové aplikace

Webových aplikací pro překlad textů můžeme najít celou řadu. Na první pohled je u většiny z nich zřejmé, že nejsou určeny pro profesionální překlad. Uživatelské rozhraní tomu neodpovídá, i kvalita překladu je mnohdy na velmi nízké úrovni. Pro překlad většího souvislého textu se z dostupných aplikací hodí například Google Translator Toolkit[5]. Jedná se o freewarový nástroj, který podporuje importování několika běžných typů formátů. Uživatel si vybere zdrojový jazyk a jazyk, do kterého se má dokument přeložit a nástroj poté ze sdílené databáze přeloží jednotlivé věty.



Obr. 5. Uživatelské rozhraní online nástroje Google GTT

V případě, že se jedná o některý z běžných jazyků, u kterého existuje rozsáhlá databáze, dostaneme uživatel již přeložený dokument. V opačném případě překládaná část obsahuje částečně přeložený dokument, kde se vyskytují přeložené i nepřeložené věty. Nedostatky této aplikace jsou například:

- Google GTT nepřekládá kontextově.
- Aplikace se snaží přeložit všechna slova, tzn. i obchodní značky nebo názvy, které zůstávají shodné s originálem.
- Překládací okno může obsahovat formátovací značky, které mohou být při překlada vymazány nebo pozměněny, což vede k vygenerování nesprávného přeloženého dokumentu.
- Pokud překládací okno obsahuje větší množství chyb nebo nevhodných překladů, uživatel musí tyto chyby opravit, to může vést ke zpomalení celé práce.

Navzdory zmíněným nedostatkům lze aplikaci použít pro jednorázové projekty, u kterých neočekáváme znovupoužití vlastních přeložených segmentů a spokojíme se se sdílenou databází, která je k dispozici.

1.3 Vyhodnocení analýzy

Z uvedeného průzkumu je vidět, že existuje velký rozdíl mezi překládacími nástroji. Pokud bychom hledali kvalitní nástroj a nepotřebovali bychom sdílet databázi přeložených výrazů, určitě sáhneme po nějakém desktopovém řešení. Většina nástrojů tohoto typu nabízí jednoduché ovládání s několika málo okny, kde každé z nich má svou úlohu. Některé překladače bývají vybaveny slovníkem, jiné zase mají *translation memory*, kterou si každý překladatel buduje sám. Zásadní nevýhodou je instalace aplikace, která pro svůj běh potřebuje dostatečně výkonný hardware a také skutečnost, že se jedná o placené produkty.

Co se týče online nástrojů, ty trochu zůstávají pozadu za desktopovými aplikacemi. Na Internetu sice najdeme spoustu nástrojů, ale už na první pohled je patrné, že nejsou určeny pro delší práci. Většinou obsahují jedno okno pro vložení celého textu a překlad je zobrazen někde na stránce častokrát s nemožností editace. Navíc aplikace nabízejí z větší části jen strojový překlad, který je v některých jazycích skoro nečitelný. Chybí tu možnost překládat text po částech tak, jak jsme zvyklí z desktopových aplikací a také náhled na přeložený text. Jak je vidět, zatím žádný z nástrojů nenabízí přijatelnou kvalitu překladu nebo vhodné uživatelské rozhraní.

2 WEBOVÉ APLIKACE

Webová aplikace je aplikace typu klient-server, kdy se na straně serveru vykonává kód a na straně klienta se zobrazí výsledná stránka ve webovém prohlížeči. Nejčastěji tyto aplikace spolupracují s nějakou databází, která je uložena na serveru. Serverová část se stará o veškerou logiku programu a přístupy do databáze, na klientovi se pouze zobrazuje obsah stránky nejčastěji ve standardním formátu HTML, který podporují všechny webové prohlížeče.

Existuje několik možností, jak strukturovat webovou aplikaci[6]. Možné typy architektur jsou např.

- Bez použití vrstev – vše se nachází v jednom souboru
- Dvouvrstvá architektura – odděluje prezentační část od zbytku aplikace
- Třívrstvá architektura
 - prezentační vrstva – je zastoupena webovým prohlížečem
 - aplikační vrstva – stará se o logiku aplikace
 - databázová vrstva – komunikuje s databází
- Vícevrstvá architektura – dělení na více vrstev

Webových technologií je celá řada a každá z nich má své výhody i nevýhody. Pro prezentační vrstvu se nejčastěji volí značkový jazyk spolu s kaskádovými styly, pro druhou vrstvu pak některý ze serverových skriptovacích jazyků (PHP, ASP a další).

Kromě tradičních technologií se v poslední době používají i méně známé technologie. Jedná se většinou o open-source projekty. V této práci byly použity následující technologie: šablony pro klientkou část jsou napsány v jazyce XHTML s využitím CSS stylů, samotný obsah stránky je generovaný pomocí klientského skriptovacího jazyka JavaScriptu s využitím knihovny Backbone.js. Serverová část je napsána ve skriptovacím jazyce Node.js.

V následujících kapitolách se proto zaměříme zejména na tyto technologie.

2.1 Klientská část

Jak již bylo zmíněno výše, klientská část je napsána v jazyce JavaScript využívající knihovnu Backbone.js. Tato knihovna je postavená na architektuře MVC (Model-View-Controller), která odděluje datový model (Model) od uživatelského rozhraní (View). Třetí komponenta (Controller) pak obvykle představuje hlavní logiku aplikace na klientské straně, zajišťuje uživatelské vstupy a řídí koordinaci mezi uživatelským rozhraním a modelem. Architektura MVC byla původně navržena norským profesorem informatiky a počítačovým odborníkem Trygvem M. H. Reenskaugem v roce 1979 při práci na jazyce Smalltalk-80[7].

2.1.1 Model v Backbone.js

Datové modely spravují data v aplikaci. Nezabývají se uživatelským rozhraním, ani nepředstavují prezentační vrstvu, ale místo toho reprezentují strukturu dat, která aplikace může potřebovat. Když se změní model, typicky dojde k upozornění, že nastala změna a ostatní vrstvy mohou podle toho zareagovat[7].

Komponenta model představuje seznam atributů a jejich hodnoty. V Backbone je možné při definování modelu nastavit některým nebo všem atributům defaultní hodnoty. V případě, že po získání dat ze serveru některý z atributů chybí a je definován v defaultní části modelu, doplní se tento atribut a jeho hodnota podle příslušné definice. Zde je ukázka jednoduchého modelu, který definuje několi defaultních atributů a jejich hodnot. Kromě těchto vlastností může model po vyžádání dat ze serveru obsahovat i další atributy, které jsou uloženy v databázi.

```
var Person = Backbone.Model.extend({
  defaults: {
    name: 'Thomas',
    age: 0,
    children: []
  },
  initialize: function(){
    console.log("Welcome to this world!");
  }
});

var person = new Person();
```

Funkce `initialize()` se volá, kdykoliv se vytváří nová instance modelu. Tato funkce není povinná, volá se automaticky, ale doporučuje se napsat si vlastní verzi této funkce[8].

Hodnoty atributů i samotné atributy se nastavují pomocí funkce `set()` nebo v konstruktoru při vytváření instance objektu.

```
// creating new model
var mary = new Person({name: 'Mary', age: 21, children: ['Thomas',
'Ryan']});

// setting attributes
var patty = new Person();
patty.set({name: 'Patty'});
```

K atributům se přistupuje pomocí funkce `get()`.

```
var name = mary.get('name'); // Mary
var age = mary.get('age'); // 21
var children = mary.get('children'); // Thomas, Ryan
```

Aplikace může obsahovat skupinu stejných modelů. V Backbone se tyto skupiny nazývají kolekce (Collections). Aplikační logika se v takovém případě píše pro celou kolekci modelů, ne pro samostatný model. Tímto se vyhneme psaní vlastní logiky pro každou instanci modelu.

Kolekce musí obsahovat referenci na model. Můžeme také definovat vlastnost `url`, která odkazuje na místo na serveru, kde jsou uložena data pro daný model. Získání dat se pak provádí zavoláním funkce `fetch()` nad danou kolekcí. Je to základní funkce a má za účel získat data ze serveru a přiřadit je do kolekce. Při volání funkce můžeme definovat také argumenty podobně jako například u technologie ajax. Server vrací data jako pole modelů ve formátu JSON².

```
var peopleCollection = new People();
peopleCollection.fetch({
  success: function() {
    // some code here
  },
  error: function(err) {
    console.log(err);
  }
});
```

² JSON (JavaScript Object Notation) – univerzální datový formát nezávislý na platformě, snadno čitelný. Představuje kolekci párů název/hodnota

2.1.2 View v Backbone.js

Uživatelské rozhraní je vizuální reprezentace modelů a obsahuje již filtrovaná a formátovaná data. Obecná myšlenka je organizování celé webové stránky do logických celků obsahující jednotlivá view (pohledy), přičemž každá tato část může být modifikována nezávisle na ostatních. Jedna z výhod tohoto přístupu je, že při změně jedné části se na klientovi překreslí pouze modifikovaná oblast. Nedochází tak k neustálému překreslování celé stránky při každé změně.

Views v Backbone neobsahují formátovací značky, ale slouží spíše jako podpora pro definování logiky modelu, která nám říká, jak mají být data zobrazena uživateli. Obvykle se k tomu používá některý z JavaScriptových templatovacích nástrojů (např. Mustache, jQuerytmpl)[7].

Vytváření nového view je víceméně přímočaré. Jednotlivé pohledy se implementují pomocí funkce `Backbone.View.extend()`. Základním prvkem každého view je klíčové slovo `el`, který představuje referenci na DOM element. Existují dva způsoby, jak přiřadit DOM element do prvku `el` ve view: DOM element již existuje nebo programátor manuálně vytvoří nový element. Pokud tento prvek není specifikovaný, Backbone vytvoří svůj vlastní objekt, kterým je prázdný `div` element.

Můžeme si vybrat z několika typů templatovacích frameworků. Mezi nejznámější patří Handlebars.js a Underscore.js. Šablona bývá napsána v HTML jazyce a obsahuje proměnné, za které se při generování obsahu stránky dosazují hodnoty z modelu.

Následuje ukázka šablony v Underscore.js.

```
<script type="text/html" id="template-contact">
  <div class='contact'>
    <strong><%= name %></strong>
    <span><%= email %></span>
  </div>
</script>
```

V JavaScriptu poté definujeme odkaz na tuto šablonu pomocí atributu `id` takto:

```
template = _.template($("#template-contact").html());
```

Generování obsahu stránky se spustí zavoláním funkce `render()` při vytváření objektu view. Tato funkce není povinná, definuje logiku pro generování šablony a je volána v inicializační části.

```
var ContactView = Backbone.View.extend({
  template: _.template($("#template-contact").html()),

  render: function() {
    // This is a dictionary object of the attributes of the models.
    // => { name: "Jason", email: "j.smith@gmail.com" }
    var dict = this.model.toJSON();

    // Pass this object onto the template function.
    // This returns an HTML string.
    var html = this.template(dict);

    // Append the result to the view's element.
    $(this.el).append(html);

    // ...
  },

  initialize: function(){
    // Call function render()
    this.render();
  }
});
```

2.1.3 Controller

Kontrolery jsou prostředníkem mezi modelem a pohledem a mají klasicky dvě zodpovědnosti: jednak modifikují vyrenderované view při každé změně modelu a modifikují model, kdykoliv uživatel manipuluje s view. Na rozdíl od ostatních MVC architektur, Backbone nemá skutečný kontroler. Logiku kontroleru obstarávají Views, a pro řízení aplikace slouží Routers. Ani jeden z nich však není v pravém smyslu kontroler[7].

2.2 Serverová část

Node.js je platforma, která vychází z V8 JavaScript engine od společnosti Google, a byla navržena pro psaní rychlých, škálovatelných internetových a serverových aplikací. Programy pro Node.js jsou psané v jazyce JavaScript a používají událostmi řízený I/O model, ideální pro datově náročné real-time aplikace, které běží na všech distribuovaných zařízeních[9].

Mnoho vývojářů používá programovací jazyk JavaScript výlučně pro psaní uživatelského rozhraní webových stránek. Node.js nám dovoluje, aby tento populární programovací jazyk mohl být aplikován v širším kontextu, zvláště pak na webových serverech[10].

Node obsahuje více než 6 000 modulů, které představují sadu funkcí a jsou komukoliv dostupné. Jelikož se jedná o poměrně mladý projekt, Node.js má silnou vývojářskou

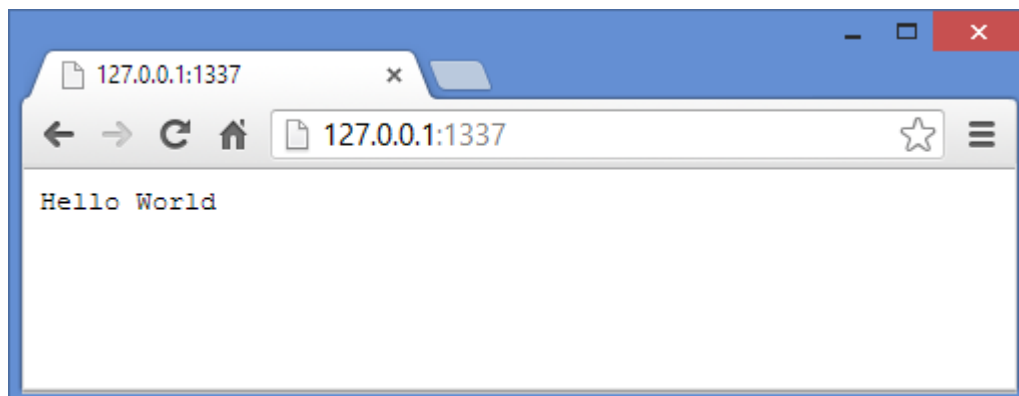
komunitu, která se podílí na vytváření těchto modulů[11]. Při psaní aplikací velice často programátoři spoléhají právě na tyto moduly.

Jeden z hlavních cílů Node.js je poskytování vysoce škálovatelného serverového prostředí. Pro vytvoření serveru se používá modul HTTP, který poskytuje rychlou implementaci HTTP serveru. Na rozdíl od ostatních jazyků, jako je např. PHP, které běží uvnitř serveru Apache, Node sám představuje webový server běžící na lokálním prostředí a proto je nutné jej nejdříve vytvořit[10]. Následuje ukázka implementace jednoduchého HTTP serveru:

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, "127.0.0.1");
console.log('Server running at http://127.0.0.1:1337/');
```

Server naslouchá na portu číslo 1337 na adrese `http://127.0.0.1:1337/`. Pro zobrazení obsahu stránky stačí napsat `http://127.0.0.1:1337/` do url prohlížeče.



Obr. 6. Zobrazení obsahu stránky ve webovém prohlížeči

2.3 Databáze

Druhá (střední) vrstva aplikace je napsaná v programovacím jazyce Node.js, o které jsme mluvili v předchozí kapitole. Tento jazyk podporuje několik typů databází, z nichž si představíme databázi MongoDB.

MongoDB (od slova „humongous“) je open-source dokumentově orientovaný databázový systém. Je součástí NoSQL databází a je napsán v jazyce C++[12]. Na rozdíl od relačních

databází, Mongo ukládá data ve struktuře s dynamickým schématem nazvané BSON³, což je zkratka Binary JSON. Stejně jako JSON, BSON podporuje vkládání objektů a polí dovnitř dalších objektů. Databázové záznamy se nazývají dokumenty, databázové tabulky pak kolekce. MongoDB automaticky každému dokumentu přiřazuje unikátní pole `_id`, které zastává funkci primárního klíče. `_id` je typu *ObjectId*, což je 12-bitový BSON datový typ.

V Node je čtení a zápis dat v tomto formátu velmi efektivní. Mongo podporuje JavaScriptové funkce uvnitř dotazů, tímto se stává velmi mocným nástrojem pro čtení, včetně redukování potřebných dotazů[10].

Node má obrovskou podporu pro Mongo prostřednictvím své vlastní knihovny Mongoose. Ve srovnání s ostatními nativními drivery, Mongoose je prostředí, které dovoluje více intuitivní vytváření modelů a schémat[10].

2.3.1 Schema

V MongoDB se datové schema definuje jinak než u relačních databází. V relačních databázích je potřeba nejdříve vytvořit databázové tabulky a následně můžeme ukládat data. Každá entita v tabulce je úzce spojena s položkou v datovém modelu. Datový model je vytvářen odděleně od samotného programu. Naproti tomu MongoDB nepožaduje explicitní definování struktury pro ukládaná data. Ve skutečnosti nemusí mít žádný datový model, ten se vytváří až při ukládání samotných dat. Můžeme přidat novou vlastnost (v relační databázi bychom to nazvali entitu) do tabulky i několik měsíců po začátku práce s aplikací, aniž bychom předefinovali předchozí schéma a ovlivnili tak již uložené záznamy[10].

Následující příklad ukazuje, jak definovat jednoduché schema v Mongoose.

```
var mongoose = require('mongoose')

var Schema = mongoose.Schema,
    ObjectId = Schema.ObjectId

var AuthorSchema = new Schema({
  name: {
```

³ binární reprezentace datového formátu JSON – zdroj <http://bsonspec.org/>

```
    first : String,
    last : String
  },
  contact: String,
  photo : String
});

var CommentSchema = new Schema({
  commenter : String,
  body : String,
  posted : Date
});

var ArticleSchema = new Schema({
  author : ObjectId,
  title : String,
  contents : String,
  published : Date,
  comments : [CommentSchema]
});

var Author = mongoose.model('Author', AuthorSchema);
var Article = mongoose.model('Article', ArticleSchema);
```

Vytvoření databáze a připojení se děje následujícím způsobem:

```
mongoose.connect( 'mongodb://localhost/article-database' );
```

2.3.2 Vyhledávání

Získávání dat z kolekce se provádí pomocí metody `find()`. Syntaxe knihovny Mongoose dovoluje řetězení podmínek, podle kterých se mají záznamy filtrovat. Dotaz se provádí zavoláním metody `exec()` s návratovou hodnotou. Všechny dokumenty z jedné kolekce můžeme vyhledat pomocí jednoduchého dotazu, jak je uvedeno níže:

```
Article.find().exec(function(error, result) {
  if(!error) {
    // handle result
  } else {
    // error handling
  }
});
```

U složitějších dotazů můžeme využít řetězení podmínek a vyžádat si data z databáze jediným dotazem. Například budeme chtít najít 10 nejnovějších článků. V Mongoose bude mít dotaz následující tvar:

```
Article.find().sort({published: -1}).limit(10)
  .exec(function(error, result) {
    if(!error) {
      // handle result
    } else {
      // error handling
    }
  });
```

Hodnota `-1` u vlastnosti `published` znamená setřídění záznamů sestupně podle vlastnosti `published`. Pokud bychom chtěli setřídít záznamy vzestupně, tj. od nejstaršího po nejnovější, použili bychom zápis `Article.find().sort({published: 1})...`

2.3.3 CRUD operace

CRUD (Create-Read-Update-Delete) jsou čtyři základní operace používané databázovými stroji.

Create operace vytváří nový záznam v dokumentu nebo kolekci. MongoDB má pro tuto operaci dvě metody, `insert()` a `save()`. `insert()` je primární metoda pro vložení nového záznamu do kolekce. Metoda `save()` potom provádí vložení nového záznamu, pokud záznam neobsahuje vlastnost `_id`, jinak provádí modifikování záznamu s tímto `_id`. Následuje příklad uložení nového dokumentu do databáze:

```
var newAuthor = new AuthorSchema({
  name: {
    first : "Jane",
    last  : "Austen"
  },
  contact: "mail@example.com",
  photo  : "photo.png"
});

newAuthor.save(function(error) {
  if(!error) {
    // handle success
  } else {
    // handling error
  }
});
```

Operaci *Read* jsme si uvedli v předchozí kapitole. Kromě metody `find()` má MongoDB i Mongoose zaimplementovanou metodu `findOne()`, která vrací první dokument v definované kolekci.

Operace *Update* je reprezentována metodou `update()`. Tato metoda akceptuje příznak „`upsert`“, který modifikuje chování metody. Možnosti jsou následující:

- Pokud dotaz nalezne existující dokument, metoda se chová jako *update*.
- Pokud dotaz nenalezne žádný existující dokument, metoda se chová jako *insert*.

```
Author.update({first: "Jane", last: "Austen"},
              {$set: {last: "Lastausten"}}).exec(function(error) {
  if(!error) {
```

```
    // handle success
  } else {
    // handling error
  }
});
```

Pro odstranění záznamu z databáze slouží metoda `remove()`.

```
Author.remove({_id: id}).exec(function(error) {
  if(!error) {
    // handle success removing
  } else {
    // handling error
  }
});
```

2.4 Shrnutí

V předchozích kapitolách jsme si stručně představili technologie, které se začínají prosazovat vedle klasických technologií a stále častěji se používají pro vytváření webových aplikací. Jsou to převážně nové technologie a stále se ještě vyvíjejí. Kolem nich existuje celá komunita odborníků i nadšenců, kteří se podílejí na vývoji. Společným rysem je rychlejší vývoj aplikace, použití jednoho programovacího jazyka (JavaScriptu) pro psaní jak serverové, tak i klientské části, nezávislost na platformě a další.

II. PRAKTICKÁ ČÁST

3 CÍLE APLIKACE

Cílem této práce je vytvořit webovou aplikaci, která bude sloužit pro překládání dokumentů. Bude zaměřena především na překlad většího souvislého textu s využitím uživatelské databáze.

3.1 Požadavky

Aplikace bude poskytovat vhodné uživatelské rozhraní pro práci s textem. Uživatel si nejprve nahraje soubor v některém z podporovaných formátů, který chce přeložit a poté nástroj rozdělí dokument na jednotlivé segmenty. Segmenty se ukládají do databáze, aby se mohly znovu použít. Překladači se během práce zobrazují všechny potřebné informace, jako je zdrojový text, překládaný text i nápověda v podobě přeložených segmentů. Ukládání přeložených segmentů se bude dít automaticky nebo po stisknutí tlačítka Save (Uložit).

3.1.1 Funkční požadavky

Aplikace bude splňovat následující požadavky:

- Aplikace si bude udržovat informace o stavu každého souboru
- Aplikace bude podporovat ukládání zdrojových i přeložených segmentů do databáze
- Aplikace bude schopna na základě podobnosti vyhledat v databázi odpovídající přeložené segmenty
- Aplikace bude poskytovat náhled zdrojového i přeloženého dokumentu

3.1.2 Nefunkční požadavky

- Bude se jednat o webovou aplikaci
- Aplikace bude poskytovat jednoduché a intuitivní uživatelské rozhraní
- Bezproblémový chod v podporovaných prohlížečích. Podporované prohlížeče: Internet Explorer 9+, Chrome, Firefox

3.2 Návrh databáze

MongoDB je jednou z předních NoSQL databází, které se dnes hojně využívá především pro svou škálovatelnost. Databáze pracuje s dynamickým schématem, tzn. že není třeba definovat pevnou strukturu dat. Aplikace obsahuje několik kolekcí, které si postupně představíme. MongoDB se podstatně liší od SQL databází tím, že neobsahuje tabulky, proto místo tabulek budou v práci použité definice schématů jednotlivých kolekcí. Všechna schémata jsou zapsána v souboru `db.js`.

3.2.1 User

Kolekce **User** má následující strukturu:

```
var User = new mongoose.Schema({
  username: { type: String, required: true, index: { unique: true } },
  password: { type: String, required: true },
  email: { type: String, required: true }
});
```

Účel této kolekce je zřejmý. Vlastnost `username` je datového typu `String` a obsahuje uživatelské jméno. U této vlastnosti je definován atribut `unique`, který zajišťuje unikátní výskyt uživatelského jména v celé kolekci. Vlastnost `password` obsahuje uživatelské heslo v šifrované podobě. Šifrování se provádí pomocí funkce `hashSync(password, salt1, null)` modulu `bcrypt-nodejs` při ukládání nového uživatele do databáze, kdy se nejprve definuje parametr `salt1` (tj. délka pole náhodných bytů) a z hesla se vypočítá hash a ten se následně uloží do databáze. Při autentikaci uživatele se zavolá funkce `compareSync(password, passwordFromDb)`, která zkontroluje vypočítaný hash s uloženým hashem uživatele.

3.2.2 File

```
var File = new mongoose.Schema({
  user: { type: String, required: true },
  fileName: String,
  customName: String,
  source: String,
  target: String,
  type: String,
  status: String,
  created: Date,
  updated: Date
});
```

Kolekce **File** obsahuje informace o souboru, který si uživatel nahraje. Vlastnost `user` obsahuje `username` (uživatelské jméno) uživatele. Vlastnost `fileName` pak název souboru. Pro lepší orientaci v souborech kolekce obsahuje vlastnost `customName`, která není povinná. Uživatel si může zadat u každého souboru vlastní pojmenování. Vlastnosti `source` a `target` se vztahují k jazykům, `source` definuje zdrojový jazyk a `target` cílový jazyk. Vlastnost `type` určuje formát souboru, např. `xml`. Vlastnost `status` znamená stav překládaného dokumentu. Po uložení nového dokumentu na server je `status` nastavený na `New`. Pokud uživatel začne na dokumentu pracovat, změní se hodnota na `InProgress`. Po ukončení práce a označení dokumentu v uživatelském rozhraní jako `Complete` (dokončený), změní se `status` na `Completed`. Jestliže uživatel chce i nadále pokračovat v překladu dokumentu, může ho tlačítkem `Open` (Otevřít) opět vrátit do stavu `InProgress` a `status` se nastaví na `InProgress`. Vlastnost `created` zaznamenává datum uložení dokumentu na server, vlastnost `updated` pak datum poslední změny.

3.2.3 FileSegment

```
var FileSegment = new mongoose.Schema({
  file: String,
  sourceValue: String,
  targetValue: String,
  position: Number
});
```

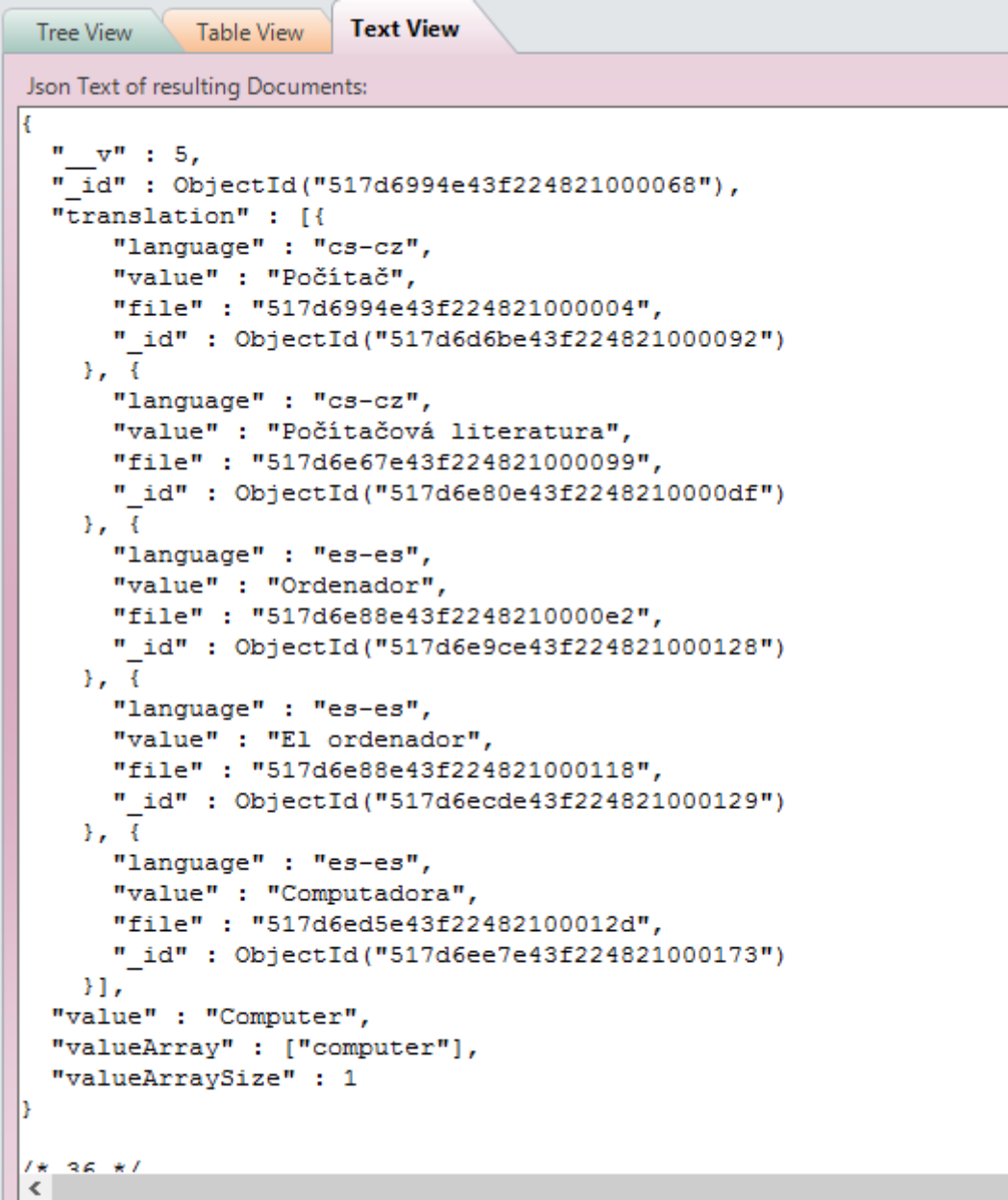
Když uživatel v uživatelském rozhraní klikne na tlačítko `Translate`, aplikace načte obsah souboru, podle typu souboru vybere parser, vytvoří segmenty a ty poté uloží do kolekce **FileSegment**. Tato kolekce obsahuje vlastnost `file`, kde se nachází `_id` souboru, tedy odkaz na soubor, v kterém se daný segment vyskytuje. Vlastnost `sourceValue` obsahuje výraz ve zdrojovém dokumentu, vlastnost `targetValue` pak výraz v přeloženém dokumentu, pokud existuje. Vlastnost `position` je datového typu `Number` a obsahuje pořadí, kde se v daném dokumentu nachází.

3.2.4 SegmentDatabase

```
var SegmentDatabase = new mongoose.Schema({
  value: {type: String, index: {unique: true}},
  valueArray: [],
  valueArraySize: Number,
  translation: [{
    language: String,
    value: String,
    file: String
  }]
```

```
} ]  
});
```

Kolekce **SegmentDatabase** je stěžejní kolekcí v celé databázi. Obsahuje seznam všech dosud překládaných segmentů a jejich překladů do různých jazyků. Tato kolekce slouží pro vyhledávání přeložených výrazů na základě překládaného jazyka. Vlastnost `value` obsahuje zdrojový segment. V kolekci je unikátní, aby se zabránilo duplicitním hodnotám. Dále kolekce obsahuje pomocné vlastnosti `valueArray` a `valueArraySize`, kam se ukládají jednotlivá slova ze segmentu pro snazší vyhledávání. Vlastnost `translation` pak obsahuje pole subdokumentů, které se skládají z vlastností `language`, kam se ukládá kód cílového jazyka, `value`, kde se nachází překlad rodičovského segmentu v daném jazyce a vlastnosti `file`, tj. v kterém souboru byl daný překlad použitý. Jako příklad si můžeme uvést dokument, který je znázorněný na obrázku (Obr. č.7). Zde je vidět, že zdrojovému segmentu „Computer“ odpovídá pět různých překladů (subdokumentů). Dva z nich mají kód jazyka „cs-cz“ (čeština) a jejich překlady jsou „Počítač“ a „Počítačová literatura“. Ostatní mají kód jazyka „es-es“ (španělština) a překlady „Ordenador“, „El ordenador“ a „Computadora“. Jestliže nějaký překládaný dokument bude obsahovat stejný zdrojový segment a překlad, který ještě v databázi neexistuje, uloží se překlad k tomuto segmentu jako další subdokument.



```

{
  "_v" : 5,
  "_id" : ObjectId("517d6994e43f224821000068"),
  "translation" : [{
    "language" : "cs-cz",
    "value" : "Počítač",
    "file" : "517d6994e43f224821000004",
    "_id" : ObjectId("517d6d6be43f224821000092")
  }, {
    "language" : "cs-cz",
    "value" : "Počítačová literatura",
    "file" : "517d6e67e43f224821000099",
    "_id" : ObjectId("517d6e80e43f2248210000df")
  }, {
    "language" : "es-es",
    "value" : "Ordenador",
    "file" : "517d6e88e43f2248210000e2",
    "_id" : ObjectId("517d6e9ce43f224821000128")
  }, {
    "language" : "es-es",
    "value" : "El ordenador",
    "file" : "517d6e88e43f224821000118",
    "_id" : ObjectId("517d6ecde43f224821000129")
  }, {
    "language" : "es-es",
    "value" : "Computadora",
    "file" : "517d6ed5e43f22482100012d",
    "_id" : ObjectId("517d6ee7e43f224821000173")
  }
],
  "value" : "Computer",
  "valueArray" : ["computer"],
  "valueArraySize" : 1
}
/* 36 */

```

Obr. 7. JSON reprezentace jednoho dokumentu kolekce SegmentDatabase v aplikaci MongoVUE

3.2.5 Language

```

var Language = new mongoose.Schema({
  name: { type: String, required: true },
  codeLong: { type: String, required: true, index: { unique: true }
});

```

Kolekce **Language** obsahuje seznam jazyků a jejich kódů. Vlastnost `codeLong` je unikátní, aby nedocházelo k duplicitním záznamům. Uživateli se v aplikaci zobrazí název jazyka současně s kódem jazyka.

3.3 Architektura aplikace

3.3.1 Struktura aplikace

V kořenovém adresáři jsou umístěné následující soubory:

- `app.js` – definuje potřebné moduly, funkce pro vytvoření serveru, konfigurace serveru, routes a příkaz pro spuštění aplikace.
- `db.js` – obsahuje schémata databáze pro MongoDB.
- `mustache.js` – soubor napsaný v jazyce JavaScript, který slouží pro generování obsahu html stránky. V aplikaci se používá výjmečně.

Kořenový adresář obsahuje tyto podadresáře:

- `node_modules` – moduly, které se používají v aplikaci.
- `public` – soubory s kaskádovými styly, obrázky a JavaScripty.
- `routes` – soubory, které obsahují logiku aplikace.
- `uploads` – nahrané soubory pro překlad. Název souboru na serveru se shoduje s `_id` v kolekci `File`.
- `views` – šablony v html jazyce.

3.3.2 Serverová část aplikace

Soubor `app.js`

Konfigurace serveru je zapsána v souboru `app.js`. Aplikace používá k vytvoření serveru modul `express`, který poskytuje robustní sadu různých funkcí pro vytváření webových aplikací[13]. Samotné vytvoření se provádí zavoláním funkce `createServer()`.

V souboru jsou definované reference na všechny potřebné moduly a ostatní soubory v aplikaci. Následuje seznam použitých modulů:

```
var application_root = __dirname, // kořenový adresář
    express = require("express"), // Web framework pro vytvoření serveru
    path = require("path"), // modul pro práci s cestami k souborům
    mongoose = require("mongoose"), // MongoDB databáze
    mustache = require("./mustache.js"), // JavaScriptový soubor pro
renderování html šablony
    fs = require('fs'); // modul pro čtení a zápis souborů
```

Nastavení serveru se provádí pomocí funkce `app.configure()`, která obsahuje metody `app.set()` a několik metod `app.use()`. Tyto metody zajišťují zavolání všech potřebných funkcí ihned po vytvoření serveru.

Soubor obsahuje také seznam všech HTTP dotazů a referencí na jejich implementaci v externím souboru.

Aplikace se spouští zavoláním následující funkce:

```
app.listen(8000, function () {
    console.log("Express server listening on port %d in %s mode",
app.address().port, app.settings.env);
});
```

Soubor db.js

Soubor `db.js` obsahuje definice schématů databáze tak, jak bylo uvedeno v kapitole 3.2.

Navázání spojení s lokální databází se provádí příkazem `mongoose.connect('mongodb://localhost/cat-database')`.

Soubor index.js

V podadresáři `routes` se nacházejí ostatní soubory serverové části. Soubor `index.js` má několik funkcí. Po spuštění aplikace a provedení autentikace uživatele se vytvoří objekt `store`, který slouží pro uchovávání statických dat v aplikaci. Do vytvořeného objektu `store` se ukládají informace o uživateli a seznam jazyků. V průběhu práce se sem ukládají a aktualizují informace o souboru, na kterém uživatel právě pracuje.

V souboru `index.js` jsou také definované reference na potřebné moduly a ostatní javascriptové soubory. Soubor obsahuje funkce pro renderování obsahu všech webových stránek. Funkce, které implementují logiku aplikace a přístupy do databáze, jsou pro větší přehlednost napsané v externích souborech, v `index.js` se nachází pouze jejich volání s potřebnými parametry. Výjimku tvoří funkce `loadLanguages()`, která se volá při spuštění aplikace a má za úkol načíst seznam jazyků z databáze a výsledek uložit do

objektu `store`, a funkce `loadFileInfo()`, která načítá z databáze informace o právě překládaném souboru.

Soubor user.js

Úkolem funkcí v souboru `user.js` je vytvoření nového uživatele a autentikace uživatele na základě uživatelského jména a hesla. Obsahuje tyto funkce:

- `newUser` – uložení nového uživatele do databáze. Heslo je před uložením zašifrováno pomocí funkcí z modulu `bcrypt-nodejs`⁴.
- `changePassword` – funkce modifikuje uživatelské heslo.
- `authenticate` – zde dochází ke kontrole uživatelského hesla s uloženým hashem v databázi. Po úspěšné autentikaci je uživatel přesměrován na hlavní stránku aplikace.

Soubor file.js

Účel funkcí v souboru `file.js` je uložení souboru na server do adresáře `uploads`, uložení informací o souboru do kolekce `File` a vyhledávání informací o souboru podle zadaných kritérií. Obsahuje následující funkce:

- `newFile` – uloží do databáze informace o novém souboru.
- `saveFile` – přiřadí souboru název podle `_id` z databáze a uloží jej na server do adresáře `uploads`.
- `userFiles` – vyhledá v databázi informace o všech souborech daného uživatele.
- `updateFileInfo` – aktualizuje informace o souboru v databázi.
- `readFile` – načte obsah souboru do paměti a vrátí jeho obsah ve znakové sadě `utf-8`.

Soubor segment.js

Úkolem funkcí v tomto souboru je načíst obsah souboru do paměti a podle typu souboru vybrat parser, vytvořit segmenty a uložit je do kolekce `FileSegment`. K tomu slouží následující funkce:

⁴ <https://github.com/shaneGirish/bcrypt-nodejs>

- createSegment – načte obsah souboru a zavolá funkci segmentFile.
- segmentFile – podle typu souboru zvolí příslušný parser.
- saveSegment – zde se vytvoří objekt segment a uloží se do kolekce FileSegment.
- readXml – funkce pro parsování xml dokumentů. Zde se volají funkce z modulu xmldom⁵. Jedná se o JavaScriptovou implementaci W3C DOM a aplikace jej využívá pro vytvoří objektu s DOM strukturou.
- xmlRecursive – pomocná funkce, která prochází vytvořený DOM objekt a hodnoty z uzlů typu text ukládá do databáze.
- readHtml – funkce pro parsování html dokumentů, využívající modul htmlparser⁶.
- readHtmlRecursive – obdoba funkce xmlRecursive, rekurzivně prochází DOM objekt a hodnoty uzlů typu text ukládá do databáze.
- readTxt – funkce rozdělí text po větách a tyto věty uloží do databáze. K rozdělení textu se používají regulární výrazy.
- writeFile – na základě zdrojového textu a přeložených segmentů se vytvoří přeložený dokument. Pokud některé segmenty v databázi neobsahují přeložený výraz, použije se zdrojový.
- getSourceSegments – funkce vyhledá v databázi zdrojové segmenty k danému souboru.
- updateSegment – funkce uloží nebo modifikuje hodnotu v poli targetValue, která si uchovává přeložený výraz.

Soubor translationMemory.js

Úkolem funkcí v tomto souboru je uložit k danému zdrojovému segmentu v kolekci SegmentDatabase subdokument obsahující jazyk a překlad a následné vyhledávání subdokumentů k zadanému jazyku. Soubor obsahuje následující funkce:

⁵ <https://github.com/jindw/xmldom>

⁶ <https://github.com/tautologistics/node-htmlparser>

- saveSegment – z databáze si načte informace o zdrojovém segmentu a zavolá funkci addTranslation.
- addTranslation – nejprve se projde seznam všech subdokumentů k danému zdrojovému segmentu a pokud se v daném jazyce zadaný překlad v žádném subdokumentu nevyskytuje, vytvoří se nový objekt, který se uloží jako další subdokument ke zdrojovému segmentu.
- getSegments – vyhledá překlady podle jazyka a zadaných slov.

4 UŽIVATELSKÉ ROZHŘANÍ APLIKACE

Uživatel bude nejčastěji využívat hlavní stránku, na které se nachází seznam všech jeho souborů a dále pak stránku pro překládání dokumentů.

4.1 Registrování a přihlášení do aplikace

Po spuštění aplikace se uživateli zobrazí logovací stránka. Zde vyplní své jméno a heslo. Po ověření na straně serveru je pak uživatel přesměrován na hlavní stránku.

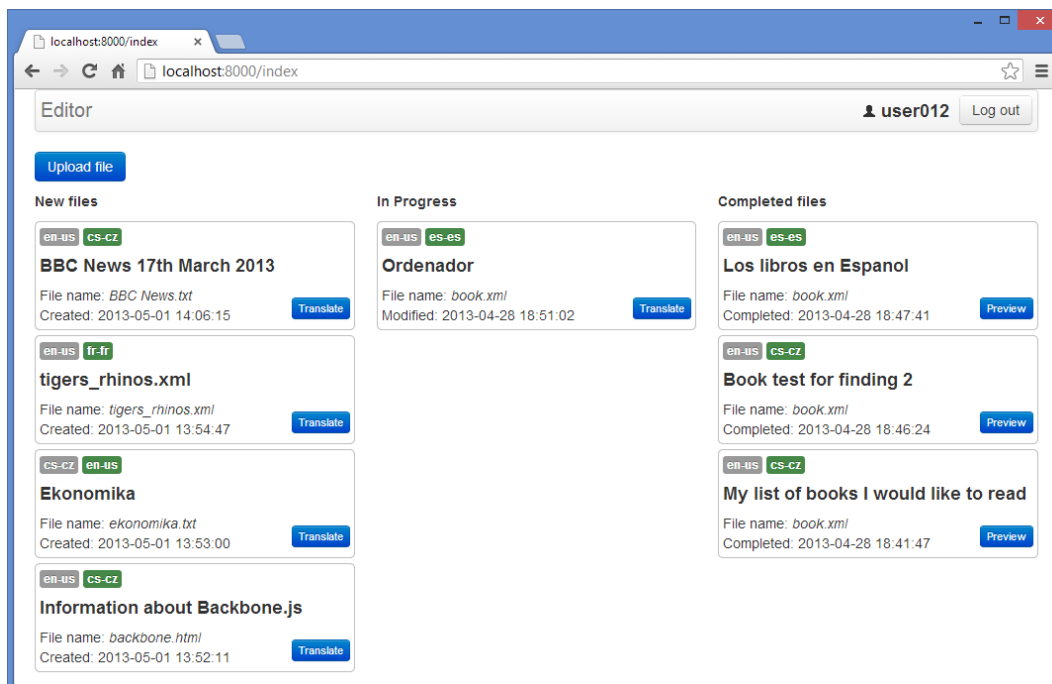
Pro nového uživatele je k dispozici link Register now! (Registrovat nyní!), který zobrazí registrační formulář. Po vyplnění všech polí a odeslání formuláře se nový uživatel uloží do databáze a zobrazí se mu hlavní stránka. Validace všech polí probíhá pomocí JavaScriptu na straně klienta a následně pak i na serveru.

4.2 Hlavní stránka aplikace

Po úspěšném přihlášení do aplikace je uživateli zobrazena hlavní stránka aplikace. Každá stránka obsahuje navigační lištu se jménem uživatele a tlačítkem pro odhlášení z aplikace.

V horní části stránky je tlačítko Upload file (Nahrát soubor), které uživatele přesměruje na stránku obsahující formulář pro nahrání souboru na server.

Na hlavní stránce je zobrazen přehled o všech souborech uživatele. Stránka je členěna do tří sekcí podle stavů dokumentů. V první sekci nazvané New files (Nové soubory) jsou zařazené soubory, které si uživatel uložil, ale ještě na nich nezačal pracovat. V druhé sekci nazvané In Progress (Rozpracované) jsou zobrazené soubory, na kterých uživatel pracuje. V poslední části nazvané Completed files (Dokončené soubory) uživatel nalezne seznam souborů, které si označil jako Completed (Hotové). Základní informace o každém souboru jsou zobrazeny ve formě kartiček v příslušné sekci. Kromě informací o jazycích, názvu souboru a datumu se na kartičce nachází tlačítko pro vykonání akce podle stavu souboru. Pro soubory v prvních dvou sekcích má tlačítko název Translate (Přeložit) a po stisknutí je uživatel přesměrován na stránku, kde se mu zobrazí segmenty daného dokumentu a může s nimi dále pracovat. Tlačítko v poslední sekci má název Preview (Náhled) a po stisknutí se uživateli zobrazí stránka s náhledem zdrojového a přeloženého dokumentu.

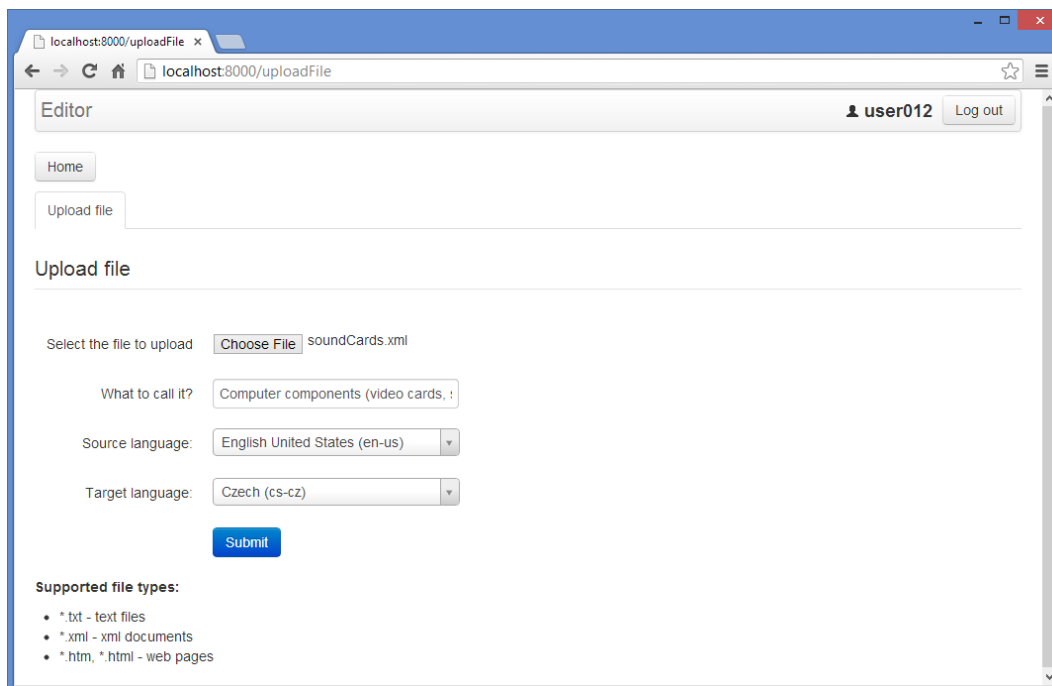


Obr. 8. Hlavní stránka aplikace

4.3 Stránka pro nahrávání souborů

Na stránku pro nahrávání souborů na server se uživatel dostane z hlavní stránky po kliknutí na tlačítko Upload files. Na této stránce se v horním rohu nachází tlačítko Home (Domů) pro návrat na hlavní stránku. Pod ním je naobrazen formulář, který obsahuje komponentu pro nahrání souboru, textové pole pro zadání názvu a dva rozbalovací seznamy s jazyky. První seznam slouží k výběru zdrojového jazyka, druhý pak cílového jazyka. Pole pro vyplnění názvu souboru není povinné, slouží pro vlastní pojmenování souboru. Soubor se uloží na server po stisknutí tlačítka Submit (Odeslat).

V dolní části se pak nachází seznam podporovaných formátů souborů.




Obr. 9. Stránka pro nahrání souborů

4.4 Překladová stránka

Po kliknutí na tlačítko Translate, které se nachází na kartičkách v prvních dvou sekcích, se uživateli zobrazí překladová stránka. Nahoře je umístěn panel s tlačítkem Home, které vrátí uživatele na hlavní stránku, na pravé straně jsou zobrazena tlačítka Complete (Hotový), Save (Uložit) a Save & Close (Uložit a zavřít). Význam těchto tlačítek je popsán dále v této kapitole.

Pod tímto panelem jsou zobrazeny informace o souboru, jako je název souboru, z jakého jazyka a do jakého jazyka se daný dokument překládá, na pravé straně pak status dokumentu a datum poslední změny.

Zbytek stránky zabírají záložky Translation (Překlad) a Preview (Náhled). První, aktivní záložka je rozdělená do tří částí. V první části nazvané Source sentences (Zdrojové věty) je seznam všech segmentů, které dokument obsahuje, v druhé části Target sentences (Přeložené věty) je textové pole pro psaní překladu a v poslední části Translation Memory se pak zobrazují překlady, které odpovídají překládanému segmentu. Kliknutím na jednotlivé segmenty v části Source sentences se aktivní segment zvýrazní a v pravé části se zobrazí ručička  s nápovědou Copy source (Kopíruj zdroj). Aby uživatel nemusel opisovat text, který odpovídá zdrojovému textu, může kliknout na tento symbol a zdrojový

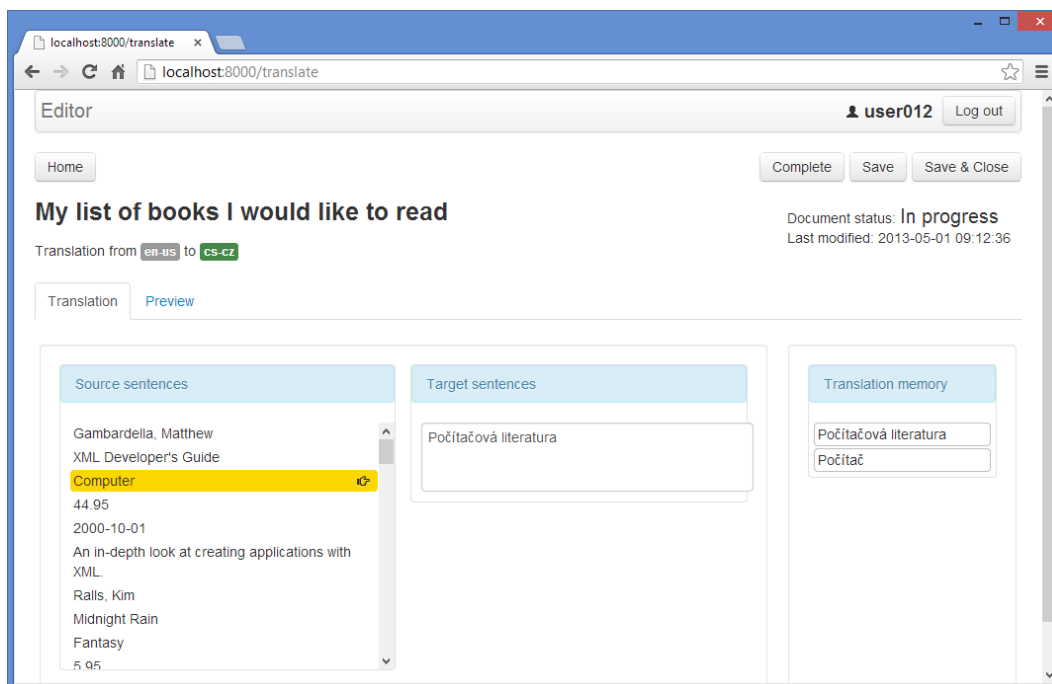
text se zkopíruje do druhé části. Jestliže k vybranému segmentu již existuje překlad v databázi, zobrazí se v druhé části jeho překlad. Zároveň se z databáze načtou již existující překlady k tomuto segmentu a zobrazí se v poslední části. Po kliknutí na některou z těchto položek se nabízený překlad zkopíruje do textového okna v prostřední části.

Druhá záložka nazvaná Preview ukazuje uživateli náhled na zdrojový a překládaný dokument. Obsahuje dvě okna, Source document (Zdrojový dokument) a Target document (Přeložený dokument).

Uložení aktuálního překladu může uživatel provést kliknutím na tlačítko Save. Nicméně každá překládaná položka se automaticky ukládá po kliknutí na jinou položku v části Source sentences.

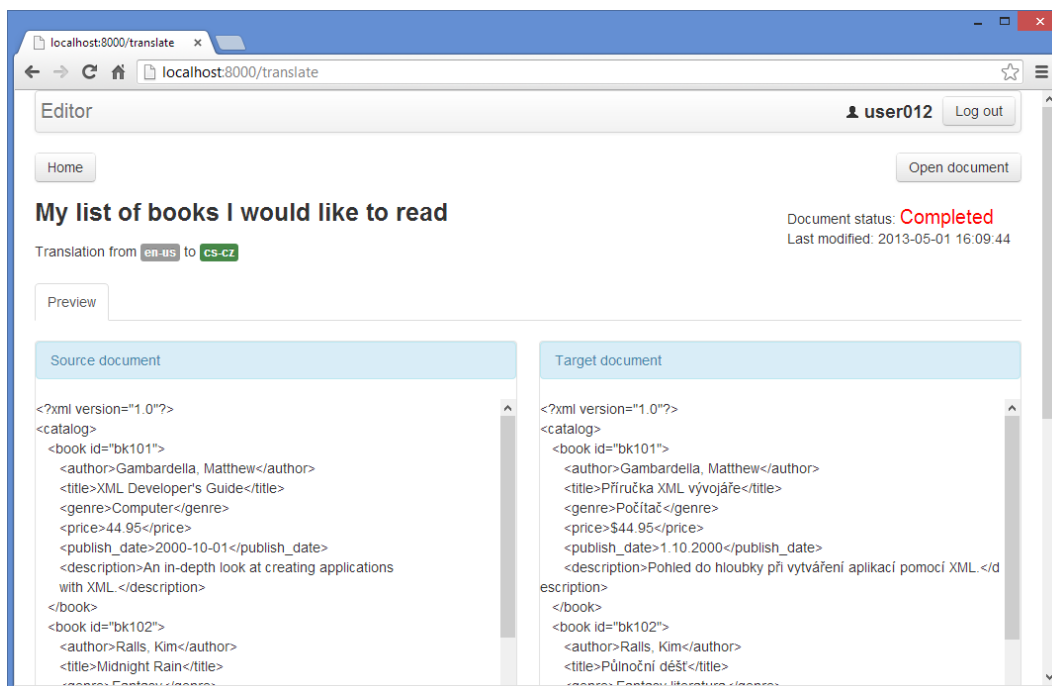
Tlačítko Save & Close slouží pro uložení aktuálního překladu a přesměrování na hlavní stránku aplikace.

Jakmile uživatel dokončí překlad dokumentu, může kliknutím na tlačítko Complete změnit jeho stav. Do databáze se uloží tento nový stav a stránka se znovu načte s aktuálními informacemi. Na stránce se pak zobrazí jen záložka Preview.



Obr. 10. Překladová stránka

Pokud si uživatel zobrazí již hotový dokument (na hlavní stránce klikne na kartičku v posledním sloupci), na překladové stránce bude mít k dispozici pouze náhled zdrojového a přeloženého dokumentu. Na panelu s tlačítky je teď zobrazeno jen tlačítko Open document (Otevřít dokument), který umožňuje vrátit dokument do stavu In Progress. Tím se uživateli zpřístupní také záložka Translation a může dále pokračovat v překladu dokumentu.



Obr. 11. Náhled zdrojového a přeloženého dokumentu

5 DALŠÍ ROZŠÍŘENÍ

Aplikace v současném stavu nabízí jen základní funkcionalitu, tj. nainportování souboru pro překlad, zobrazení zdrojových a přeložených vět, práci s databází. Dalším krokem k zdokonalení nástroje je implementace připojení uživatelských slovníků. Například open-source slovníky Hunspell distribuované pod licencí GPL nabízejí kromě vyhledávání slov také kontrolu pravopisu, morfologický analyzátor (analýza zohledňuje skloňování a časování slov v daném jazyce), navíc obsahují podporu pro integraci pro většinu platforem a programovacích jazyků. Jelikož většina překladatelů pracuje s nástroji jako je TRADOS, bylo by užitečné zaimplementovat také připojení těchto uživatelských *translation memory*. Dalším možným rozšířením je přidání komponenty pro vložení překladového textu pomocí techniky Copy Paste a našlo by se toho mnohem více.

6 SYSTÉMOVÉ POŽADAVKY

Pro testování je možné aplikaci používat v lokálním prostředí. Pro vytvoření serveru je potřeba nainstalovat Node.js a pro ukládání dat databázi MongoDB. Oba tyto produkty běží na různých platformách a podporují 32-bitové i 64-bitové architektury. MongoDB pro Windows požaduje operační systém novější než Windows XP.

6.1 Instalace Node.js

Poslední verze je ke stažení na stránce <http://nodejs.org/download>. Po nainstalování se aplikace spouští z příkazové řádky pomocí příkazu `node app.js`. Více v kapitole 6.3.

6.2 Instalace MongoDB

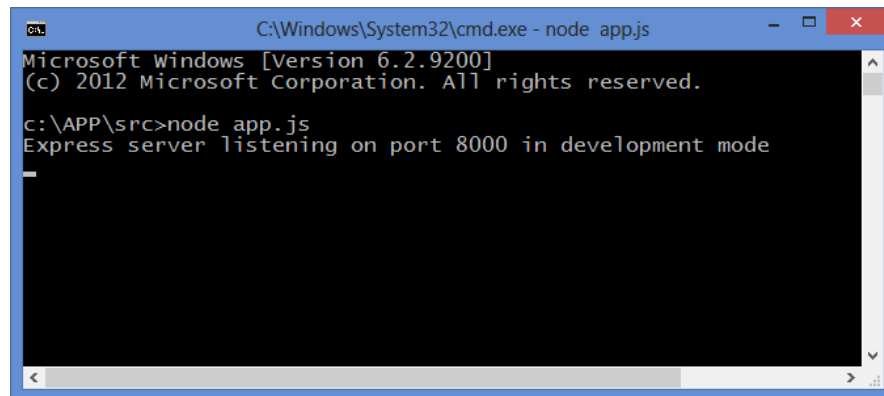
Instalační balíčky pro databázi MongoDB jsou ke stažení na stránkách mongoDB <http://www.mongodb.org/downloads>.

Konfigurace databáze se provádí pomocí příkazového řádku. Manuál pro každou platformu je k dispozici na stránce <http://docs.mongodb.org/manual/installation/> nebo v dokumentu [MongoDB-Manual.pdf](#) v sekci **Installation Guides**.

MongoDB pro Windows požaduje nastavení MongoDB jako Windows Service. Postup konfigurace je popsán v sekci **MongoDB as a Windows Service**.

6.3 Spuštění aplikace

Adresář *src* z příloženého CD zkopírujte na disk. Otevřete konzoli, přejděte do adresáře *src* a napište příkaz `node app.js`. Po chvíli se na konzoli objeví zpráva, že server byl vytvořený a je aktivní.

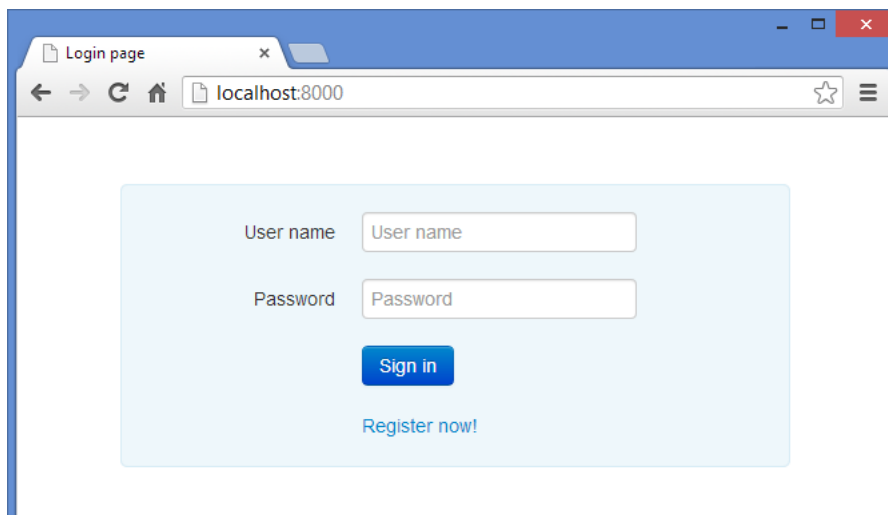


```
C:\Windows\System32\cmd.exe - node app.js
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

c:\APP\src>node app.js
Express server listening on port 8000 in development mode
```

Obr. 12. Konzole po spuštění aplikace

Otevřete prohlížeč a do url napište adresu `http://localhost:8000`. Zobrazí se přihlašovací stránka.



Obr. 13. Přihlašovací stránka

ZÁVĚR

Cílem této práce bylo vytvořit online aplikaci pro překlad většího souvislého textu do různých jazyků. Práce je zaměřená na základní funkcionalitu nástroje a předpokládá se další rozšíření tak, aby uživateli nabízel především pohodlnou práci při překládání.

V teoretické části je provedena analýza dostupných překládacích nástrojů. U desktopových aplikací bychom neměli problém si vybrat kvalitní nástroj, nevýhodou těchto řešení je jen to, že se převážně jedná o komerční produkty. Online aplikací je na trhu celá řada, ale řešení mnohých z nich je nedostačující po všech stránkách. Většinou jsou vhodné pouze pro základní porozumění přeloženého textu. Z tohoto důvodu byl vznesen požadavek na vytvoření online aplikace, která by v budoucnu představovala kvalitní nástroj s příjemným uživatelským rozhraním, bohatou slovní zásobou a mnoha dalšími funkcemi pro práci s textem.

Druhá část teoretické části je zaměřená na popis technologií, které byly použity pro vývoj aplikace. Pro ukládání dat se ukázal jako nejvhodnější databázový systém MongoDB, který nepožaduje pevnou strukturu dat, ale dynamicky si vytváří objekty a ty pak ukládá do databáze. Pro programování logiky aplikace a konfigurace serveru byl použitý skriptovací jazyk Node.js a generování obsahu html stránek je psáno v JavaScriptu za pomoci knihovny Backbone.js. Obě tyto technologie jsou nezávislé na platformě. Pro vývoj aplikace byl použitý vývojový open source nástroj NetBeans IDE, vytvořený především pro vývoj aplikací v jazycích Java a JavaScript.

Praktická část se zabývá rozborem aplikace. Nejprve je uveden seznam základních požadavků. Pak následuje návrh databáze s popisem jednotlivých kolekcí a jejich účelem v systému. Serverová část obsahuje seznam jednotlivých souborů a popis funkcí a jejich význam. Praktická část je zakončená stručnou uživatelskou příručkou.

Webová aplikace klade důraz na jednoduché a intuitivní ovládání. Byly použity technologie umožňující rychlou implementaci různých rozšíření. Hlavním přínosem bude bezesporu možnost využití sdílené databáze přeložených segmentů. Překladatelé se sami podílejí na vytváření databáze, a proto dbají na kvalitu překladu, aby jim v budoucnu usnadnil práci v podobě funkční nápovědy.

CONCLUSION

The aim of this thesis was develop online application for translation the larger continuous text into various languages. Paper is focused on the basic functionality of tool and is expected many further extensions in order to be offering to users mostly comfortable work with translation.

In theoretical part there is provided analysis of available translation tools. There should not be a problem with desktop applications to choose appropriate tool, the disadvantage of this solutions is the fact that tools are mainly for commercial purposes. We can find many online applications, however the solution of many of them is inappropriate in all aspects. Mostly there are suitable only for basic understanding of translated text. For this reason the requirement to develop online application was established. In the future this could represent a powerful tool with comfortable user interface, rich vocabulary and many other functionality for working with text.

The second part of theoretical part is focused on description web technologies that were used for application development. MongoDB database system has turned out as the most suitable for data storage. This system does not require fixed data structure, on the contrary it creates objects dynamicaly and these ones stores to database. For programming application logic and server configuration was used script language Node.js and rendering of html content is written in JavaScript supported by Backbone.js library. Both these technologies are platform independent. For application development was used open source development tool NetBeans IDE, primary created for developing application in programming languages Java and JavaScript.

Practical part is concerned on analyse of application. It starts with the list of basic requirements, following by the design of database with description of collections and their purposes in the system. The server side contains the list of files and descriptions of functions and their importance. Practical part is finished off with a brief user guide.

Web application puts emphasis on a simple and intuitive user interface. There were used technologies that enable a rapid implementation of various extensions. The main benefit will be certainly the possibility of using the shared database of translated segments. Translators themselves are participating on creating database and for this reason they pay

attention the translation quality in order to make their job easier by the functional help in the future.

SEZNAM POUŽITÉ LITERATURY

- [1] [online]. [cit. 2013-04-21]. Dostupné z: <http://www.langsoft.cz/>
- [2] [online]. [cit. 2013-04-21]. Dostupné z: <http://www.trados.com/en/>
- [3] [online]. [cit. 2013-04-21]. Dostupné z: <http://kilgray.com/products/memoq>
- [4] MemoQ: integrated translation environment [online]. [cit. 2013-05-12]. Dostupné z: http://kilgray.com/files/user-guide/memoQ_QuickStartGuide_5_0_EN_0.pdf
- [5] [online]. [cit. 2013-04-21]. Dostupné z: <http://translate.google.com/toolkit/>
- [6] BRADA, Přemysl a Martin DOSTAL. Webové aplikace [online]. [cit. 2013-05-12]. Dostupné z: http://pit-plzen.cz/sites/default/files/Webove_aplikace_uvod.pdf
- [7] HUGHES-CROUCHER, Tom a Mike WILSON. Developing Backbone.js Applications: up and running. 1st ed. Sebastopol, CA: Oreilly, 2012, xiv, 184 p. ISBN 14-493-2825-3.
- [8] DAVIS, Thomas. Backbone.js Tutorials [online]. [cit. 2013-01-29]. Dostupné z: <https://leanpub.com/backbonetutorials>
- [9] [online]. [cit. 2013-04-21]. Dostupné z: <http://nodejs.org/>
- [10] HUGHES-CROUCHER, Tom a Mike WILSON. Node: up and running. 1st ed. Sebastopol, CA: O'Reilly, xiv, 184 p. ISBN 14-493-9858-8.
- [11] Node for Front-end Developers. Oreilly. ISBN 978-144-9318-833.
- [12] [online]. [cit. 2013-04-21]. Dostupné z: <http://www.mongodb.org/>
- [13] [online]. [cit. 2013-04-21]. Dostupné z: <http://expressjs.com/>

SEZNAM OBRÁZKŮ

Obr. 1. Uživatelské rozhraní softwarového nástroje PC Translator	12
Obr. 2. Translation Memory interface aplikace SDL Trados 2007.....	13
Obr. 3. Překladačové okno nástroje SDL Trados 2007	14
Obr. 4. Uživatelské rozhraní nástroje memoQ.....	15
Obr. 5. Uživatelské rozhraní online nástroje Google GTT	16
Obr. 6. Zobrazení obsahu stránky ve webovém prohlížeči.....	23
Obr. 7. JSON reprezentace jednoho dokumentu kolekce SegmentDatabase v aplikaci MongoVUE	33
Obr. 8. Hlavní stránka aplikace.....	40
Obr. 9. Stránka pro nahrání souborů.....	41
Obr. 10. Překladačová stránka.....	42
Obr. 11. Náhled zdrojového a přeloženého dokumentu	43
Obr. 12. Konzole po spuštění aplikace	46
Obr. 13. Přihlašovací stránka.....	46

SEZNAM PŘÍLOH

CD se zdrojovými kódy aplikace